```java
package Alejandro_Castellaos;

import java.io.Serial;
import java.io.Serializable;
import java.util.Objects;

public class Pokemon implements Serializable {

    @Serial
    private static final long serialVersionUID = 1L;

    private String nombre;
    private String tipo;
    private int salud;
    private int nivel;

    /**
     * Constructor de pokemons
     * @param nombre - Nombre del pokemon (String)
     * @param nivel - Nivel del pokemon (int)
     * @param salud - Salud del pokemon (int)
     * @param tipo - Tipo del pokemon (String)
     */
    public Pokemon(String nombre, int nivel, int salud, String tipo) {
        this.nombre = nombre;
        this.tipo = tipo;
        this.salud = salud;
        this.nivel = nivel;
    }


    // GETTERS Y SETTERS
    public String getNombre() {
        return nombre;
    }

    public String getTipo() {
        return tipo;
    }

```

```java
    public int getSalud() {
        return salud;
    }

    public int getNivel() {
        return nivel;
    }


    // EQUALS Y HASHCODE (PARA UTILIZAR CONJUNTOS)
    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Pokemon pokemon = (Pokemon) o;
        return getSalud() == pokemon.getSalud() && getNivel() == pokemon.getNivel() && Objects.equals(getNombre(), pokemon.getNombre()) && Objects.equals(getTipo(), pokemon.getTipo());
    }

    @Override
    public int hashCode() {
        return Objects.hash(getNombre(), getTipo(), getSalud(), getNivel());
    }


    // TO STRING
    @Override
    public String toString() {
        return "Alejandro_Castellaos.Pokemon{" +
                "nombre='" + nombre + '\'' +
                ", tipo='" + tipo + '\'' +
                ", salud=" + salud +
                ", nivel=" + nivel +
                "}\n";
    }
}

```

```java
package Alejandro_Castellaos;

import java.io.*;
import java.util.*;

public class Main_Pokemon {
    public static void main(String[] args) {

        // Estructura de tipo conjunto
        Set<Pokemon> conjuntoPoke = new HashSet<>();

        // Archivos de origen (pokemon.csv) y destino (pokemon_ordenados.ser) con sus rutas
        File pokeFile = new File("Resources/pokemon.csv");
        File destinoFile = new File("Resources/pokemon_ordenados.ser");

        //try-for-resources que crea un Scanner del archivo origen
        // y el grupo de objetos de las clases que nos permiten guardar
        // objetos de forma serializada
        try (Scanner scf = new Scanner(pokeFile);
             FileOutputStream fos = new FileOutputStream(destinoFile);
             BufferedOutputStream bos = new BufferedOutputStream(fos);
             ObjectOutputStream oos = new ObjectOutputStream(bos)
        ) {

            // Mientras queden lineas en el .csv == Quedan pokemons
            while (scf.hasNextLine()) {
                String[] linea = scf.nextLine().split(",");

                //Comprobamos que el split haya separado 4 Strings (datos)
                // para saber si en la linea se encuentra un pokemon con formato valido
                if (linea.length == 4) {
                    conjuntoPoke.add(new Pokemon(
                            linea[0],
                            Integer.parseInt(linea[1]),
                            Integer.parseInt(linea[2]),
                            linea[3])
                    );
                }
            }

```

```
41              //Guardamos en una List de pokemons el conjunto de pokemons ordenados
42              // primero por ipo y en caso de igualdad por nombre
43
44              //Utilizamos un stream del conjunto de pokemons que ordenamos segun el
45              // criterio mencionado y luego convertimos a lista
46              List<Pokemon> listaOrdenada = conjuntoPoke.stream().sorted((o1, o2) -> {
47                  if (o1.getTipo().equals(o2.getTipo()))
48                      return o1.getNombre().compareTo(o2.getNombre());
49                  else
50                      return o1.getTipo().compareTo(o2.getTipo());
51              }).toList();
52
53
54              //Guardamos los objetos de forma serializada recorriendo la lista ordenada
55              for (Pokemon p : listaOrdenada) {
56                  oos.writeObject(p);
57              }
58
59          //Mostramos mensajes diferentes segun el error capturado
60          } catch (FileNotFoundException e) {
61              System.err.println("Archivo no encontrado:");
62              e.printStackTrace();
63          } catch (IOException e) {
64              System.err.println("Error de Input-Output:");
65              e.printStackTrace();
66          }
67
68          // ----- PRUEBAS -----
69
70          // Intentamos volver a leerlos y mostrarlos con el orden guardado
71
72  //          try (FileInputStream fis = new FileInputStream(destinoFile);
73  //               BufferedInputStream bis = new BufferedInputStream(fis);
74  //               ObjectInputStream ois = new ObjectInputStream(bis)
75  //          ) {
76  //
77  //              while (bis.available() > 0) {
78  //                  Alejandro_Castellaos.Pokemon p = (Alejandro_Castellaos.Pokemon) ois.readObject();
79  //                  System.out.print(p);
80  //              }
```

```
//
//        } catch (IOException | ClassNotFoundException e) {
//            throw new RuntimeException(e);
//        }
    }

}
```